Issue 32 | March 2011

blender art MAGAZINE

Blender learning made easy...

Spring is Sprung

Making 2D environments for the iPhone game Spires
A videogame in few logic bricks
Modeling an iPhone 4 with Modifiers
Creating Plants Using Arrays
CanTree - Online Tree Generator
Blender 2.5 Materials and Textures Cookbook

www.blenderart.org

Tree Sunset - by Bryan Tenorio.

CONTENTS

EDITOR - Gaurav Nawani
MANAGER/EDITOR - Sandra Gilbert
WEBSITE - Nam Pham
DESIGN - Gaurav Nawani

PROOFERS
Brian C. Treacy
Bruce Westfall
Daniel Hand
Daniel Mate
Henriël Veldtmann
Joshua Leung
Joshua Scotton
Kevin Braun
Mark Warren
Noah Summers
Patrick ODonnell
Phillip
Ronan Posnic
Scott Hill
Wade Bick
Valérie Hambert

WRITERS
Oscar Baecher
Alfonso Montón
Claas Eicke Kuhnen
Stephen Bates
Arnaud Couturier
Sandra Gilbert

COVER ART
Tree Sunset -by Bryan Tenorio

Upcomming Issue 33 - "Everything but the kitchen sink..."

Here is your chance to discuss/write about anything blender related that interests you. Do you have a favorite feature or technique that you are just dying to share with others? Or how about your super cool latest project.

Well now is the time to write all about it and share it with all of us. Here are some ideas to inspire you.

* favorite feature / tool: tell us all about it, how it works, how you use it.

* favorite technique or workflow

* personal / commercial projects you are working on

* favorite Blender book or site

EDITORIAL

Sandra Gilbert
Manager/Editor

"I bet you are all thinking that Spring isn't even close and you are right. But I would really like to get a jump start on Spring and being warm again"

Let me just start off with, I don't enjoy being cold. Which of course means I am not real thrilled with winter. Winter is a whole bunch of cold, snow, wet and nasty looking skies. And lucky me, this winter we seem to be receiving record amounts of snow and nastiness.

Now I know that even with our record amounts, we still are getting off easy compared to the eastern United States. They are getting buried under massive winter storms that are shutting down just about everything. And the United States isn't the only one getting snowed in, Europe and other parts of the northern hemisphere are getting more than their fair share of snow and nasty weather this year.

I am seriously looking forward to Spring and an end to all the winter unpleasantness as quickly as possible. Which I am pretty sure isn't scheduled to arrive for some time yet. In fact, last night I went outside and it was cold but fairly clear. I then wandered out about an hour later and there was an inch and a half of snow on the ground. Are you serious, MORE snow?

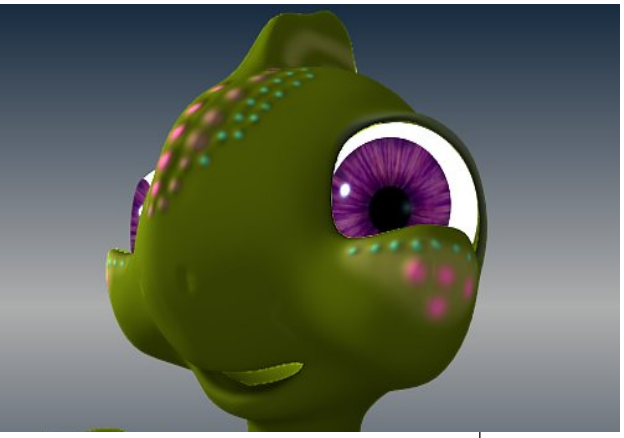
Sigh...

Which brings me to the topic of this issue of Blenderart Magazine, Spring is Sprung. I bet you are all thinking that Spring isn't even close and you are right. But I would really like to get a jump start on Spring and being warm again.

Did I mention I don't like being cold? :P

So in this issue we are going to explore some nice “Springy” topics like plants and trees. Nice warm growing things that make us think warm thoughts and forget all about being cold and snowed in. So it is time for you to settle down in your favorite reading chair and soak up some warm Blender knowledge ●

Izzy Speaks: Plant tutorials are growing



Izzy Speaks

Like anything modeled, plants can be simple and easy or highly detailed and considerably more complicated. Either way, unless you are a plant modeling genius, having a few tips can greatly ease the process. I went wandering around my favorite sites and spotted some rather nice plant inspired tutorials for you to enjoy.

BlenderCookie offers tutorials on a wide array of topics so it is no surprise that they have more than a couple of plant ones as well.

Trees

Setting up a manual version of the "Tree-from-Curves Plugin" Part 1

Setting up a manual version of the "Tree-from-Curves Plugin" Part 2

Modeling a Palm Tree

Leaves : Creating a Scene of Blowing Leaves

Landscapes and Environments to put your plants and trees in

Sculpting a Rock Face in Blender by Sketching

Creating a Simple Claymation Style Scene

And Fluffy Clouds to finish it off

Creating Volumetric Clouds

cgtuts+ has a 4 part video tutorial series on creating a rocky video game terrain in Blender. It covers how to create a finished environment for Blender's Game Engine. The main focus of the series will be texturing and lighting, with the use of some custom 2d filters to enhance the result.

Create a Rocky Video Game Terrain in Blender

Andrew Price of **BlenderGuru** has been posting some fun nature images on twitpic lately. I have noticed that anything he works on and shares generally ends up in a tutorial or tutorial series, which might indicate a nice collection of nature tutorials are in our future. If not, I think we should all poke him until he actually does make tutorials of his images. They are very nice indeed and the techniques would be a nice addition to the general education of blenderheads everywhere. (oh, make sure you poke gently please, absolutely no hurting of Blender teachers allowed. :P)

Xfrog is providing 130 free sample downloads of 3d trees and plants from its XfrogPlants collection!

You can import them into Blender using the OBJ format.

From their website:

Download free 3d plant and tree models from each of the 30 XfrogPlants libraries!

Three to nine 3d plant models are included in each of the 30 species below, and best of all, they are completely FREE! Download the plants, browse their documentation PDFs, try them in your favorite software and see firsthand our high quality modeling work. If you like them, buy and download any of the 3d libraries, instantly from our shop.

Our Botanical experts go out in the field and photograph each plant, for their references and to capture firsthand the leaf and bark textures. They then compare the plant in Nature, to established printed research work, and build 3 to 9 ages of the plant. This set of ages is required for ecosystem work – to give enough variation for instancing the models across a terrain. And if you own Xfrog 3.5, you can easily edit any of the models for additional variations !

Well that should be enough to get you on your way. After that, wander outside and observe the plants and vegetation all around you (okay you might have to wait for the snow to melt) and practice modeling what you see ●

3D WORKSHOP : Making 2D environments for the iPhone game 'Spires'

by - **Oscar Baecher**

Introduction

Spires is a story-driven fantasy Tower Defense game for the iPad and iPhone, soon to be released by Prophetic Sky Inc. It takes place in the world of Aether, a land of sprawling and colorful landscapes. It has been tamed only by Makers, powerful sorcerers who can shape their world with sheer willpower and imagination. The challenge we faced at Prophetic Sky: to make 2D sprites that expressed this creative vision, and use 3D assets in Blender to make a fun, colorful environment.

Fig1. Spires game play screenshot.

The Ground

The first part of this task was our ground textures. Spires maximized performance by breaking each level's ground into tiles, which could be recombined to make a variety of maps. On our previous game, Townrs Defender, we used Photoshop to combine two textures with various Alpha channels and export them via an Action. But this came with several limitations: we couldn't output bigger versions, and all ground textures had to use the same alpha channel, resulting in repetitive patterns. Plus, the process was highly tedious! For Spires, we realized that Blender's texture nodes would provide us with a much more flexible, scalable alternative.

So. What can you expect the process to be like?

1. Make an alpha channel

1. Make "beauty" alpha channels

2. multiply a thin black universal border

3. Add the official white borders back on

2. Use as a stencil between two ground textures on a material

3. Repeat 16 times

4. Bake on to a 16-face plane's texture

5. Render a composite to multiple outputs

To make the alpha channels, let's first quantify what we need. You will end up having 16 total textures for this first run, although in production you might make extra varieties. This covers every way you could combine Top, Left, Right, and Bottom.

For now, let's make a texture for an alpha channel that tiles on the Left and Bottom. Create a new image 256x512, name it alpha_l_b. A good naming convention will prevent lots of pain; I always name mine based on the WHITE areas of the texture, and always in the order of Left, Top, Right Bottom. Thus, alpha_l_b.png is white on the left and bottom. alpha_l_t_r_b.png would be pure white, and alpha_.png would be pure black.

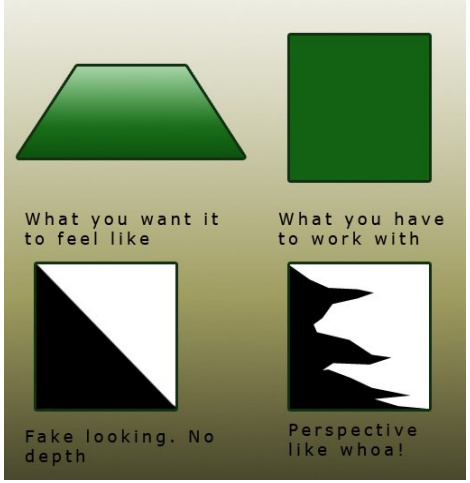


Fig2. Do's Dont's.

3D WORKSHOP : Making 2D environments for the iPhone game 'Spires'

Paint alpha_l_b.png white on the left and bottom sides, and create an organic looking transition between the corners. Here we come into some design issues: how do you represent receding 3D perspective, when you're working with squares? In a 2D sprite engine, you can't just let 3D perspective take care of things for you. This is why I start with a long texture. Later it will get scaled down, thus looking more horizontal. The nature of things receding toward a horizon means that they're longer on the X axis than they are on a Y axis, so lots of horizontal elements will help. You can also see from my example how I use extra horizontal lines to further imply this.

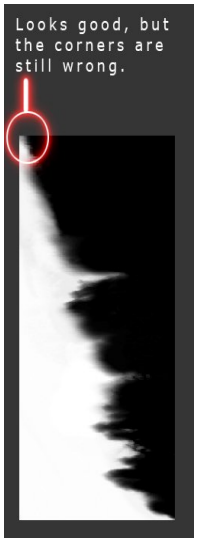
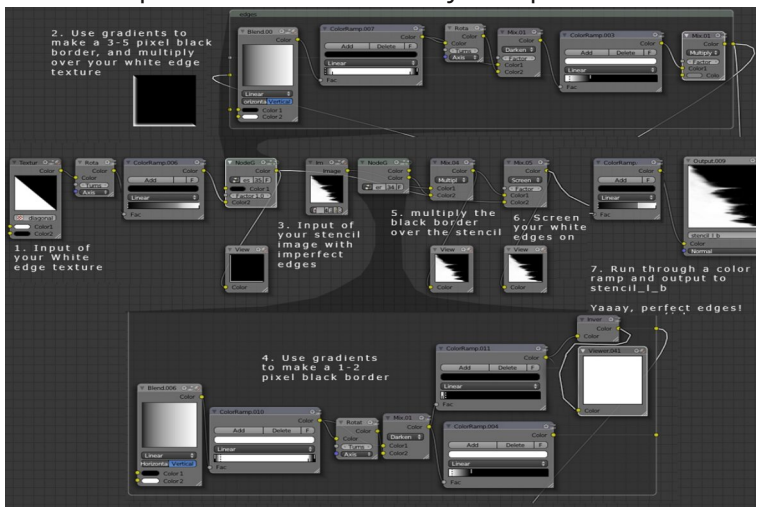


Fig3. Painted alpha

Once you're satisfied with your texture, and it's pretty accurate on its corners, go to Image \ Pack as PNG. Do this whenever you make changes, or else it won't get saved when you save your .Blend file. Now do this 16 times, once for each direction!

Create a new material with a new texture called stencil_1_8 (you're only allowed a limited number of outputs, so I split my stencils into two groups of 8), then open the texture compositor and turn on "Use Nodes." Add in the first eight textures you made; again, if your naming convention is solid, no problem knowing which ones you need. Good news! The long images conform to the size of the texture, so they're now square in size.



4. Stencil composite.

All your stencils are tiling the right way...except that they're not tiling. Minor imperfections in the texture edges will make it so you have artifacts, so next is a complex process of cleaning up our alpha edges.

We'll fix this with two steps: first, a thin black border to start from zero, and second, a white border on the edges where it needs to continue. The black border can be created just using gradients and color ramps to make a "square" gradient, then using another color ramp to remove the middle and, lastly, invert. I group this to make it less messy, and multiply it over the first alpha we drew.

To create the 16 edges, I make a diagonal blend texture and add it as an input. Why not use a blend in the compositor? By doing it as a separate texture, we can add a ramp with Black/White Position coordinates at exactly .499 and .501. This will end up giving us a perfect corner, with white on one side, black on another side, and 50% gray on the single diagonal corner pixel.

Next, use a series of Add, Rotate, and Lighten nodes to make versions of each texture. For each of these, we'll use a gradient similar to

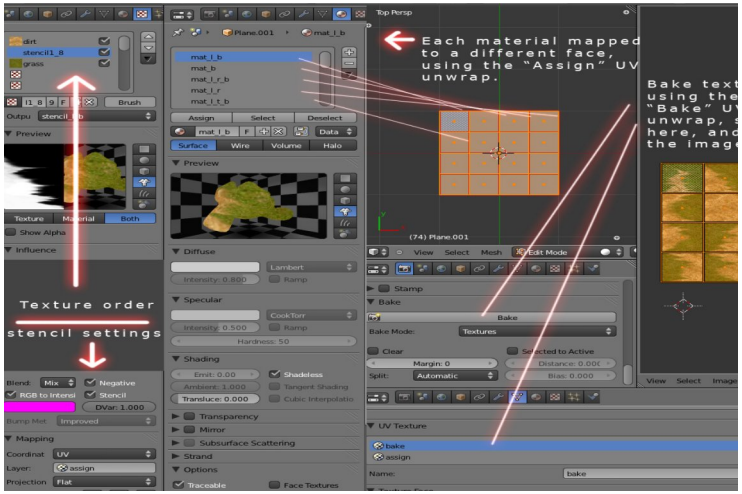


Fig5. Materials, plane, unwrap and baking.

the black border texture to cut out the middle. Make this gradient slightly thicker; we want to add over what the border just subtracted, and have it bleed into the white.

Screen this over your alpha texture. Almost done! First, run this through a simple color ramp, and then run this into an output node. The color ramp is necessary because if a Mix node is put into an output, it will retain the transfer method later. Name the output, and...do it 16 more times.

Now that the hard part is finished, we'll make a Stencil. On your material, load in two repeating ground textures, such as grass and dirt, and put the stencil layer between them. Select a Stencil output, and turn on Stencil, Negative and RGB intensity. Map everything to UV, and set the two ground textures to influence Color, the stencil influencing nothing.

Baking

Eventually you should have 16 materials, each using a different stencil output to stencil the two textures over each other. Next, create a plane and subdivide twice, so you have 16 faces. Assign a different material to each one. In the image editor, make a new 1024x1024 (4x the size of each tile) and save it. Unwrap the plane using “project from view (bounds)” and name this UV set “Bake”. Make another UV texture called “Material_assignment” and for all your textures make sure this is selected for the Mapping/ Layer assignment. For this one, the UV mapping is tricky. You want it to have all of the upper right hand verts in the upper right hand corner. So using another Project from View (Bounds) unwrap, start snapping each face's top right vert to the top right corner, etc.

Now, with Bake UVs selected and your saved 1024 texture open, go to the Render panel and bake the textures. Do they correspond to their labeling and direction? Congrats! If they don't, it is likely a problem of UVs or material assignment.

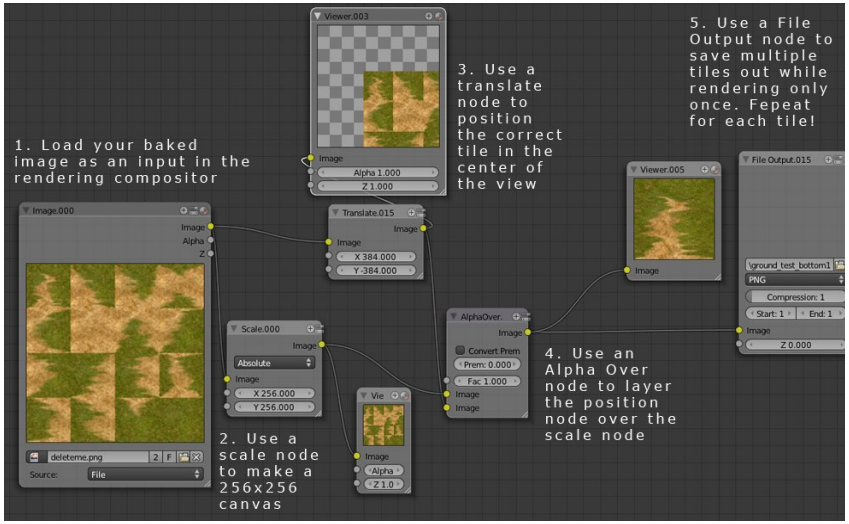


Fig6. Render output.

Final Output

Phew. Almost done. Save this image, and open up the rendering section of the Compositor. Here we'll make use of the File Output node, which will let us create a variety of renders, each to their own specifications.

Start by creating an Image input and select your baked texture. Connect it to a Scale node with Absolute X: 256, Y: 256 and set this node aside. Create a Translate node set to X: 384, Y: -384. This will put the upper left corner in the direct center of the image (use a View node to check.) Repeat for each coordinate of the texture until you have a representation of each face as the middle.

Next, take your first Translate node and connect it to an Alpha Over node, with your old Scale node as the top input. This makes it directly centered, but at the scale node's size. Finally, connect this node to a File Output node, and choose where to save it. Make sure compositing is turned on (and you've got a throwaway connected to a Composite node in the compositor) and when you render, it should spit out all your textures at once, named properly and ready to go!

Cliffs

We now have our tiles to make a 2D landscape that blends between dirt and grass. Now let's give it a nice two-level effect by

making some cliff widgets. If you look at reference photos from enough cliffs, you'll start to notice that they usually have a dominant vertical or horizontal direction; horizontal if the sedimentary layers are highly exposed, but vertical shearing with certain types of geology and environmental strain.

Because this will eventually be rendered to a 2D sprite, you have the luxury of laissez-faire poly count. I did most of the modeling by selecting multiple spaced edge loops and pulling them in, creating "sheafs" running alongside each other. Then I followed through with some extruded imperfections, rocky sediment on the cliff edge and valley, and a subtle displacement modifier with a cloud texture.

For the cliff material, add a cloud texture, set it to hard, and map it to Normal. Play with the scaling and normal strength but be careful. When working with sprites, it's very easy to get over-complicated. When you only have 100 pixels to show detail, too much will quickly become a washy, sandy blob. Add a Musgrave texture and map it to color, and use a color ramp to create a subtly varied gray look to it. Lastly, add another cloud texture, set it to voronoi crackle, influence of color with Overlay as its layer mode transfer, and add a ramp to make brown cracks over gray. This will bring more warmth into the cliffs, and also make it fit with the dirt. An important thought for environments is tying things together--what rocky material are these cliffs made of, and how does that relate to the surface sediment of the dirt?

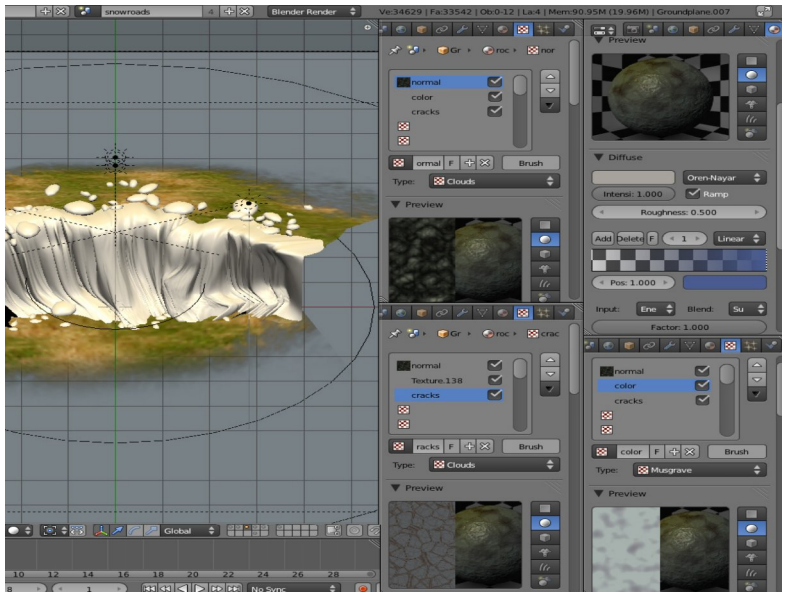


Fig7. Cliff modeling, materials and texture.

To further add some warmth back into the cliffs, on the Material settings I added a ramp to the diffuse and played around a bit. Use Alt MMB to scroll through pulldowns, particularly for your Blend mode. It takes 10 seconds to scroll through and visualize every possible blend mode, so why not do it? I ended up using a blue ramp set to Subtract and mapped to Energy, which is not something I would have expected until I saw it in thumbnail form versus other settings.

Cliff Ground

You need these cliffs to be placed on top of your ground. To accomplish this, we'll use planes with textures of the ground, but fading out toward transparency. In your image editor of choice make a texture based on your ground tiles, but fades to alpha. Create a new material with this texture set to Color and Alpha, and then in the Materials setting turn Transparency on while dragging Alpha to zero. Lastly, make the material shadeless. Model a ground plane that cleaves to your cliff edge before spreading out, and repeat for the bottom cliff edge. UV unwrap them, and position them over the blended ground texture so that they'll fade away as you leave the cliff edge.

This should now create a cliff render that can

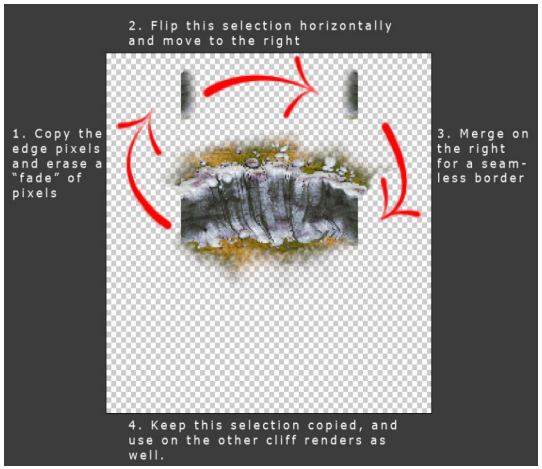


Fig8. Pixel seams.

be placed on your ground tiles without a visible break from the cliff grass. It will also create a swath of grass that breaks up the pattern of the ground below it--very handy for the thin "side" cliffs.

Rendering and Edges

To get cliffs for 8 directions, animate it rotating in 360 degrees over eight frames. Because 0 and 360 get interpreted the same, I instead key 0 degrees at frame 1 and 180 degrees at frame 2. Then go into the graph editor, select your Z rotation curve and set the interpolation to Linear. This will make the rotation continue indefinitely, and create your 360 degree turn over 8 frames.

Render it out from an orthographic camera at a 45 degree. You've now got a cliff edge from 8 different directions. But how can you get these widgets to chain together? We still need to have the line of pixels on each cliff's left and right edges to match up perfectly with any given neighbor. I found it easiest to just hand-edit this manually, so open up your image editor of choice. With your primary cliff wall select the left side, along the pixels that need to be aligned, plus some buffer pixels. Copy and paste this into a new layer, flip it and move it to the right side. Once it's lined up, gently erase the buffer pixels so that it blends smoothly together and save it.

Paste this same line of pixels (still in your clipboard) into the next cliff edge and repeat for all 8 cliffs. Now, they should be able to line up seamlessly!

Next up, try doing alternate cliffs, shorter cliffs (good for patching holes) and also ramped cliffs for characters to walk up.

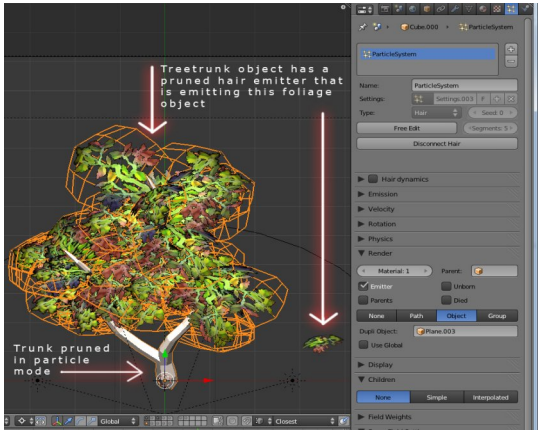


Fig9. Tree particles, alpha

Trees

We've got some nice ground with a cliff edge breaking up the landscape. But we still need some foliage to break up the monotony of this sprawling meadow. Let's make a tree widget!

I started by modeling a tree trunk with a handful of primary branches, and gave it a wood material. For the foliage, I first made a rounded plane with a single foliage alpha card mapped to it. Have the "trunk" end of the foliage start at the object's origin, as if growing outward from 0,0,0. Next, add an emitter (set to hair), set the Render type to Object and select your foliage alpha card. It will appear on the trunk as well, but don't worry about that yet. Just work on the hair settings until you have a puffy foliage look on your treetop.

Next up, switch over to Particle mode and begin erasing the hair particles on the tree trunk. Soon enough, you've got a tree with a bushy top and a rooted trunk! But you're not done yet. There's still more that needs to be done to have this tree make sense on our landscape.

Tree postprocess setup

If you placed this tree onto the background, it wouldn't look right. No shadow, and no style. So next up we're going into the render

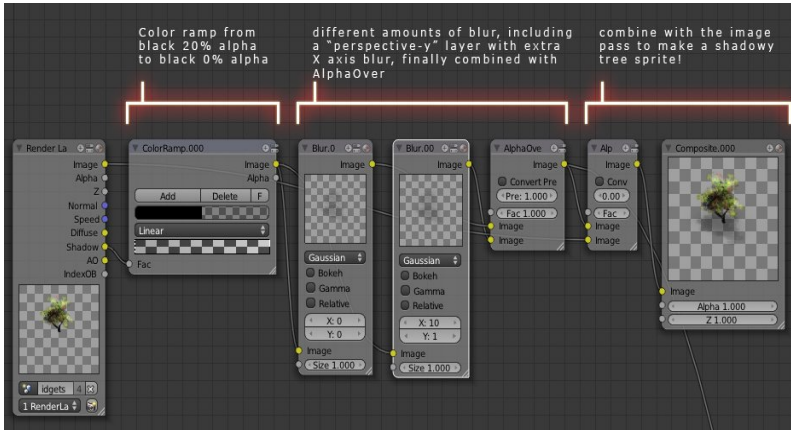


Fig. 10 Tree composite nodes.

compositor to add in several elements. This will also deal with some scene setup. Light the scene as you see fit, but have some of the lights be ray-traced. I usually go with a primary Sun light from above to cast a downward shadow, with two additional lights at angled directions to cast angled shadows.

Next, create a ground plane that's big enough to reach the camera borders. Add a new material to it. Keep Transparency turned OFF, but drag the Alpha down to zero. Under the Render Layers tab, turn on shadow pass. Render a frame and let's switch over to the rendering compositor!

Tree Compositing

My compositing philosophy, for better or worse, evolved from a mentality of replicating how I'd do things in Photoshop with nodes instead of layers. Thus, our main process will be creating several "layers" ala Photoshop or gimp, then having them stack on top of each other, occasionally with layer mode transfers. Imagine the layer stack as this, going from the bottom up: Shadow, cartoon outline, tree, occlusion, normal node lighting, and a final color correction. No prob!

If you look at the shadow pass in a viewer node, it's a white background with black for the tree's internal shadowing, and more importantly the shadow it's casting on the ground. Our goal is to get the black shadow with alpha, as if it were a layer in Photoshop, so that we can put it underneath the tree.

Feed your Shadow layer into a Color Ramp node, and hook it up to a Viewer node so you can visualize changes. On the white end of the ramp's spectrum, change the color to black and the alpha to zero. Now your viewer should show a black alpha cut-out of the tree shadow. You can adjust the alpha on the other side of the Ramp as well. I recommend this, because we want our finished shadow to be pretty subtle.

Fig 11 Elements combined

Next up you can modify the shadow further. I fed the ramp to two separate blur nodes. The first I just made slightly less noisy (1 pixel X,Y) and the second I made blurrier, especially on the X axis (X10, Y2) to make it feel scattered from the tree leaves. Feed both of these into an Alpha Over node (no premultiply conversion, no convert, premultiply and factor both at 1).

Using another AlphaOver node, composite your tree render over the shadow. Is the shadow too strong now? go back and further adjust the color ramp. Now you've got a tree sprite that you can place on the grass, and it will look like it's providing soothing shade for your game's heroes to sit under.

You can continue compositing to your heart's content. I also added a thin cartoon outline of black, ambient occlusion, and some relighting via Normal nodes to give Spires a "Saturday Morning Action Cartoon" feel.

Final Environment

So what do you get when you combine all these elements? An organic 2D sprite environment where any sense of a mechanical grid is well hidden. Take a look!

Spires is available now for the iPad, iPhone and iPod Touch for the price of $1. Acts 2 and 3 will soon be released as well. If you have any feedback on the game, we'd love to hear from you at oscar@propheticsky.com

Spires is now available for all iOS devices. You can download it at http://bit.ly/getspires. If you have a retina device, stay tuned for Spires HD which will include optimized graphics ●

3D WORKSHOP : A videogame in few logic bricks

by - **Alfonso Montón**

Introduction

This tutorial focuses on logic bricks in Blender, the only programming tool I used for Trolechaun. I gladly made this tutorial in the hopes that this will encourage someone to make nice stuff by understanding the basic structure of BGE powered games. You can download, play and see how Trolechaun was built at: http://trolechaun.weebly.com/ if you want to understand the structure you should read the tutorial first, then you would be able to get deep into Trolechaun.

For a graphic artist, logic bricks could be the bridge to understanding programming of a basic videogame. For programmers it would be a nice challenge and a good exercise to just use logic bricks.

The first thing you should know is that logic bricks have a number of limitations. However you must learn that limits are fun. If they weren't, there wouldn't be any games.

Programming a videogame is something I always wanted to do, and I could get into this using logic bricks. I could watch the results at the same time I was building Trolechaun, but I am a 3D graphic artist, not a programmer.

Problems can be solved

Limits mean Problems, but every problem can be solved, preferably with a creative solution. For example: when I made Trolechaun I thought about using mouse inputs, but I was not able to learn how to do it in time, so I just created a different game. The more skills a team has, the more solutions exist, however if you don´t know your team limits and potential you will never finish a game.

First Warning: Tidy looks better

Logic bricks can be a mess but don´t worry, you control them. Don´t forget about writing names for the Inputs and Actions, as this will save yourself a lot of confusion. Don´t forget to delete unused/unneeded logic bricks after a failed test.

Get started

Let´s say Blender logic bricks are like other software, you can paint with them in layers, but what you are painting here are actions (programmed events and animations), you can feel like a director telling his polygonized actors where you want them to move.

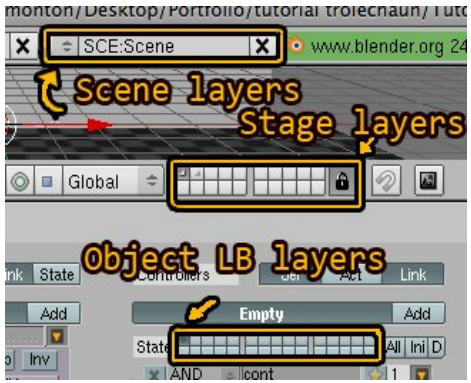
There are just three kinds of "layers" in a Blender game

Scene Layers: which contains characters, interactive objects, and any FX... these layers often contain a single "level" or stage of a game

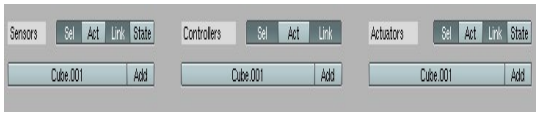
Stage Layers: Use these layers to stage your scene and actions, you decide what is visible when the game plays here.

State System (Object LB Layers): States are a way to achieve complex logics without cluttering your Logic Buttons and having to redo already existing Logic Bricks. States are a group of Logic Bricks which get executed at a certain time or state (hence the name) of the game.

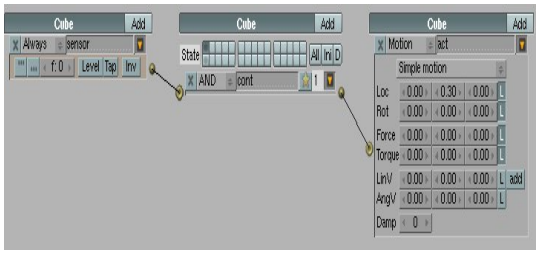
Imagine a character wandering around. There may be different materials for the ground (e.g. ice and sand), a



different control method in water or air, etc. Without states you would now start to use Properties, Property-Sensors and Actuators to control what state the actor has. With the state system probably only one LogicBrick changes the state and you are done. More information on States can be found here.

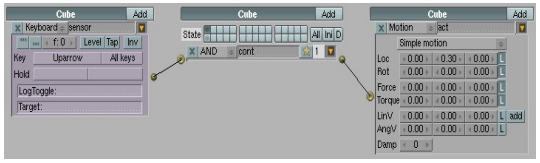


The first thing you should do when designing your game prototype is to establish the mechanics of your game and see what you can do with logic bricks. Since you are not doing a graphics showreel, characters and scenes should come after you've worked on the game mechanics.



Lets start with something easy.

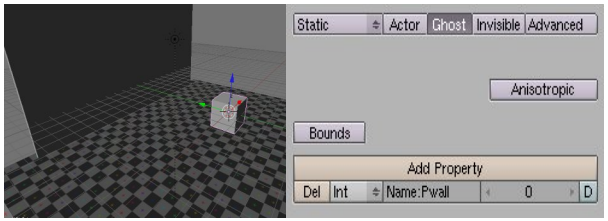
Create a cube first , then Select the cube and go to the Panels view. Get into the Logic properties and press ADD for these three: Sensor – Controller – Actuator.



Connect them and give the middle Location (Loc) variable for the Motion actuator brick a number value (0.3 in this example). Now move your mouse over the screen, press P and you will see your cube going forward automatically.

Make the cube interactive

At sensors you can see there is an actuator called Always. This means you are directing your cube to go forward. Not so difficult, right? Now change this Always sensor into Keyboard, and define UpArrow key for this sensor.



Great! Now our Cube friend is interactive. With that you can make everything move around, and that´s your basic animation resource. You can do the same with the other keys if you want your Epic Cube character to move around left, right or backwards. You can also use this method to rotate your Cube. (adjust Rot settings instead of Loc in Actuator panel).

You have to understand, all the actions in our videogame need a **Sensor** which leads an **Action**, and the connection between them are logical connectors also called **Controllers**.

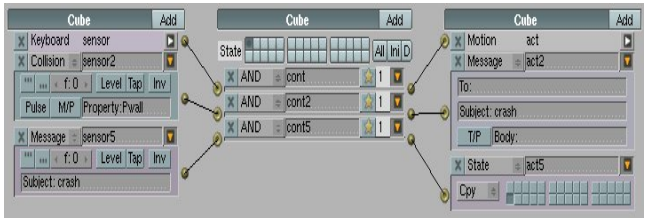
Now let's try something more complicated: Crash a cube into a Wall and then reset your position.

Scene properties

Create a wall and floor – Select the Wall and add a property to this Wall object, write Pwall as the Name of this property on the Logic Panel. Change the Physics into Static and press Ghost (if you don´t press Ghost you will see a problem with the reset re-position of our Cube character)

Object Layers

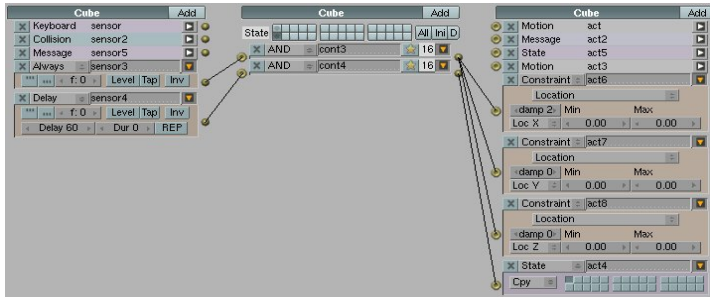
Select your cube and change the physics status Static into Dynamic (Now your cube will be in the gravity system) and create two other lines of –Sensor-Controller-Actuator. This time they will be:



Collision AND Message – Name the Collision property: Pwall (there is no difference between the Int, Bool and Timer kind of property in this case). Then name the Message Subject: crash. With that every time your character crashes into a wall, a message is sent.

3D WORKSHOP : A videogame in few logic bricks

This is very important because when you make an event, your own character receives the same message and resets. But you can use this event for other stuff (lose a life for example or creating some FX at the stage).



Message AND State - Name the Message Sensor: crash. At State you will press the second button. (This will change the state with the sensor trigger – a message is sent with this info)

With this your character will respond when it receives the same message sent by the character collision.

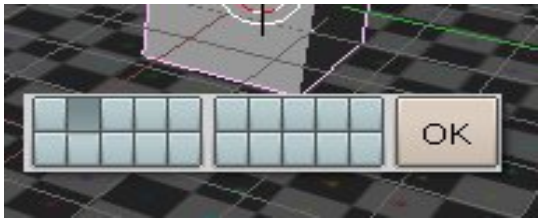
Now you can change the State of the Object logic brick at the second button and you can make different actions for the cube, you will use this time:

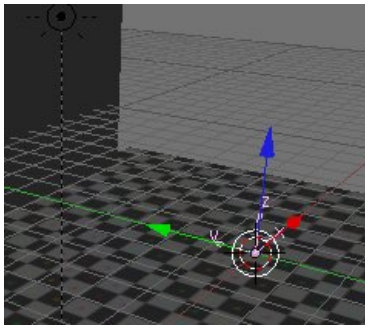
Always AND Constraint (connect with three different actuators: Constraints). These constraints will be used to define the restart position of your cube, change into Location X, Location Y and Location Z at the different Constraints.

Delay AND State – This time you write some frames of delay for this action, write 60 and at the State Actuator push the first button.

Don´t forget to come back to the first state you already programmed.

Now see what happens: Your cube crashes into the wall goes back to the first position and waits around one second (this always depends on the framerate, you can make a timer property but it would be not so simple) before it falls down and you control it again.

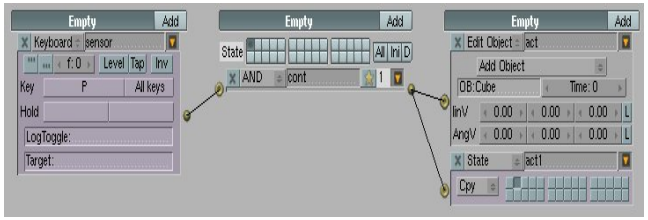




With that, you can program different states of your character, when your character crashes into a wall it should be a different state. This is why it is important to keep all your programming brick lines clean, because if you don't do this, your logic bricks will soon be a mess. So you can now control the layers of your character, now is time to control layers of your scene.

Stage Layers

When you make a game you must be tidy. When I made Trolechaun I didn´t know yet, but you have to create different layers for the stage (game environment), the character and objects and yet another layer for special effects (FX).



Now that you have your Cube Character programmed, we want to move it into the second layer. Select your Cube Character, Press M, select the second layer and your Cube will be moved to that layer of the Stage (game environment). Now you can bring your objects to the first layer (your main layer of the scene) where all the action is running.

Now create an Empty object in the stage layer. Add a Sensor-control-actuator to program it, then change the actuator from Edit Object to Add Object and write down the name of your Character in the OB field. You must write the name of your character exactly how is written on your object, with caps if it has them, otherwise the logic brick will not recognise it.

If you keep the sensor for this actuator in Always you would not notice a difference

3D WORKSHOP : A videogame in few logic bricks

when you test it. Change the sensor into a keyboard sensor and assign it a key for input. Every time you press the button you will now "receive" another character from the layer where the Cube character is stored.

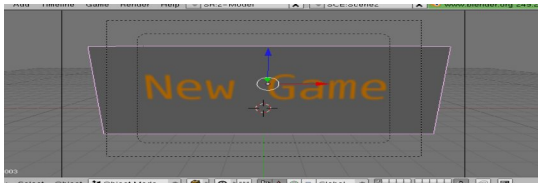
Connect a new State actuator at the same controller, so that when you press the key you've assigned, not only do you bring your character to the stage (game environment), but the state of the empty object changes into an empty state (awaiting further messages to change the state of the cube). It is easier to bring objects from other layers than to delete them from your scene.

Additionally, you use this same method when you want FX or ammo in your scene. You store all the FX and AMMO objects on another layer and use Empty objects on your main stage layer to bring them onto the stage. If they are on the same layer it is not going to work. Also you can get very interesting results when you make a chain of FX and empty objects.

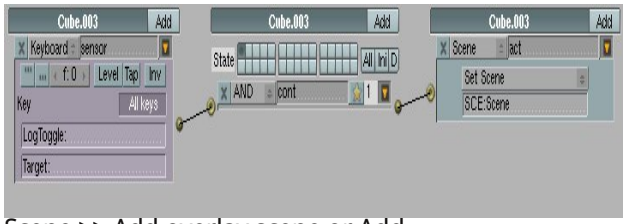
Scene Layers

Scene Layers contain information about your stage(or game level). When you create one scene you have to think about your cameras. You can have multiple cameras, you can parent your active camera to characters, make other static cameras, etc. You can also change the camera when the game is running with the actuator: Scene >> Set camera.

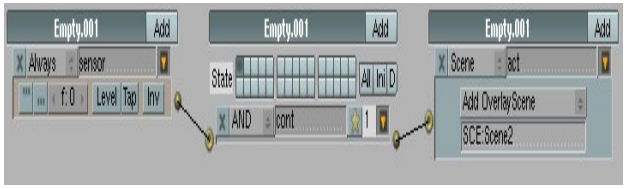
But here comes the most difficult part: How do I move all the game info like "lives left" or how many points I have between the different game scenes? Without any scripting or python programming, there is still a way. You noticed there are different properties for the Objects in the Logic Panel: Bool, Int, Timer and these properties are running just for active Scene Layer. These count the lives and points, etc.



What I used in the Trolechaun options and Game info, was to use one SCENE with an Empty object getting all the information (in this scheme Layer0). Over this Main Empty layer I was getting the visual information from the other Scene Layers with the Logic Brick:



Scene >> Add overlay scene or Add Background.



This is really important if you want to be able to pause your game, one scene must be active when you make other scene FREEZE or PAUSE. One layer (Layer 0 in this case) must be running.

Between overlayed scenes the only thing you can use are messages, because there is no better option to get information using logic bricks.

To finish the Tutorial lets create a camera, change the view into this camera and place a plane with a texture saying New Game.

You can make an action with a keyboard sensor and change the scene with the Scene actuator. Make this Scene to replace it with your previous stage Scene – (These Scenes are Game layers and they are changed by Set Scene)

3D WORKSHOP : A videogame in few logic bricks

But you ALWAYS have to start your Game from the Empty layer where you will maintain all of your game info because you will run the game at this MAIN layer. This main layer is where you have to overlay the different Scenes we created before.

Finally you have the Empty layer: layer0 with an empty object which stores the info like lives remaining and other useful stuff stored in the properties of this object. Similar to our stage layer that uses the empty object to bring the character into our scene, this empty gives the information to the HUD which is overlayed onto the game layer. The HUD layer makes visible the info you have on your Scene layer 0. You can communicate between the layers with messages as I showed you on this tutorial.

With all these basic Logic bricks and a bit of curiosity you should be able to make a Trolechaun type game yourself, this is just an introduction. Animation, illumination and shader info are things you would need to learn if you want a very well-made game and you have time or a big team. With logic bricks you can easily get a good result in a few hours.

You can get more info at http://trolechaun.weebly.com/index.html and Blender Wiki which has useful information. Also other places like Blenderartist are full of nice game designers who can make really awesome stuff ●

3D WORKSHOP : Modeling an iPhone 4 with Modifiers

by - **Claas Eicke Kuhnen**

Introduction

Modeling an iPhone 4 with all the details seems to be a more challenging task for beginners than it is actually in reality. The secret is to break down all the seemingly bigger problems into smaller more manageable elements. Then it will become quite evident how easily this model can be approached.

The key to success here is to start with very simple shapes and make use of the modifiers such as Mirror, Solidify, Screw, Bevel, and Boolean to rapidly develop shapes and proportions and then to fine tune details. This is a more parametric modeling approach borrowed from standard CAD engineering applications.

Step by step design decisions are made interactively, trying to eliminate time consuming restarts or mesh adjustments by hand, saving time and maximising design decisions.

Blender 2.5 is a vast improvement over 2.49. Besides the already mentioned new modifiers, it also allows you to now place blue prints into respective views, which eliminates the need to work in multiple view ports. By just switching between the views Blender now displays the predefined blue print. A very simple but greatly useful feature. In addition the new unit system and measuring add-ons also greatly help in working perfectly in scale.

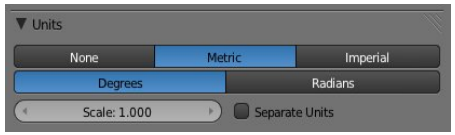
This process of using modifiers first and at the end, making the mesh into a tangible form ready for texturing is what this tutorial will focus on.

In regards to detail level, it is also very important to be clear about the modeling goals, thus gathering as much reference information as possible is very important. One needs to know what to model and those models which very often impress people are those which also pay attention to the object details.

Step 1: Scene Set-up

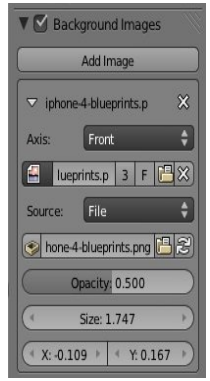
Before we start modeling, as a first step it is important to set-up the scene.

The units utilized in these blue prints are in Metric. In Blender 2.5 the Units panel is found inside the World tab. It allows the user to select between common Blender Units, the Metric or Imperial system.



As a second step, it is required to place the correct blueprint parts into the respective views.

The Background Image tab is placed inside the Properties panel and can be opened with [N] while being inside the 3D View or using the main menu 3D View>>View>>Properties.



The panel itself is quite self explanatory. It starts with the view you would like the image to visible in. It is also possible to adjust the opacity as well as size and position.

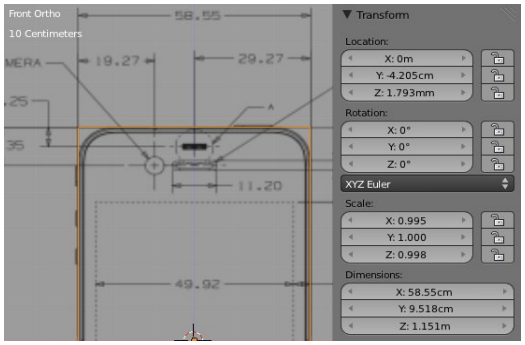
To adjust the scale of the image, a cube with the right proportions can be used as an overlay.

Through the Size and X/Y sliders the image can then be scaled and positioned correctly.

After this step the cube can be discarded as it only functioned as a scale reference.

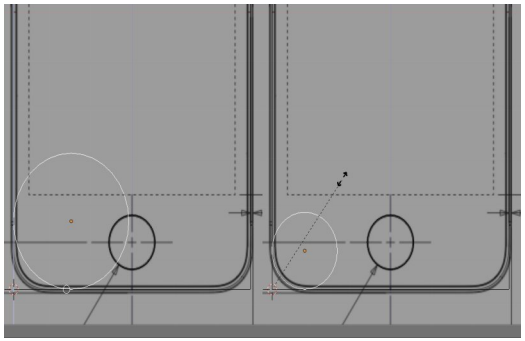
3D WORKSHOP : Modeling an iPhone 4 with Modifiers

Another nice addition is that Blender 2.5 now displays the current view name as well as the view scaled inside the upper left corner of the 3D view port.



Step 2:Blocking out basic proportions

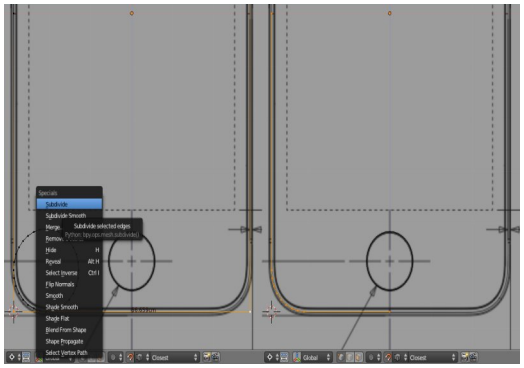
The easiest way is to start with a square perfectly placed at the center of the iPhone front view with the correct dimensions. Add



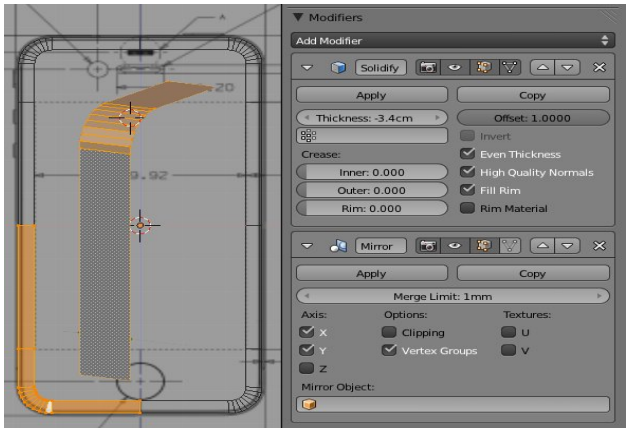
a circle and move it to the lower left corner.

The Snap to Edge function can assist in positioning the Circle to correctly lay on the lower left edge of the rectangle. Selecting the circle and pressing [g] for Grab and then [z] for a vertical movement restriction and releasing the mouse when the circle snaps to the lower edge of the square as shown on the left of the image. With the 3D cursor positioned at the lower left corner of the rectangle it can then be used to scale the circle to the right size when the Pivot Point is set to the 3D cursor as shown in the right of the image.

The following two images show how the initial square is being subdivided, the circle being joined to it and all not needed geometry being removed. The end result will be a perfect quarter of the iPhone main body.



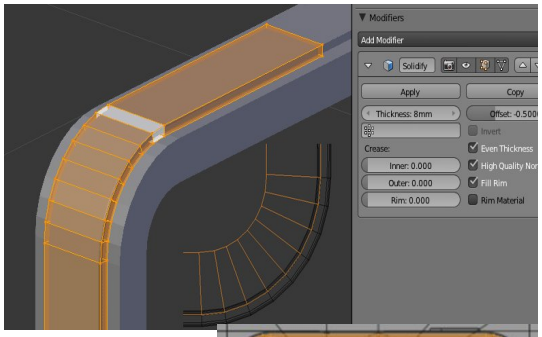
At this point this profile can be extruded into the required depth and by making use of the mirror and solidify modifier the outer border of the iPhone body is built. The mirror modifier can be used to replicate the quarter arc over the object's center as visible in the screenshot. The solidify modifier creates a thickness to the resulting surface. Because of the surface normals facing inside the offset value of "1" extrudes the depth towards the center of the object.

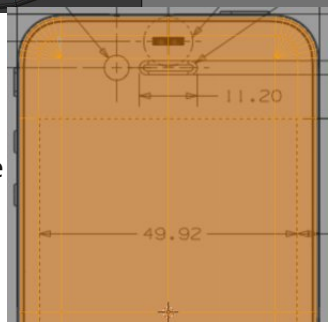


Attention was paid to provide geometry points for the future display panel mesh at this stage by using a Thickness value of "-3.4cm".

This object can be duplicated and scaled along the depth axis to build the base mesh for constructing the rim. The solidify modifier will again prove very useful in this situation. By changing the Offset to "-0.5", the modifier offsets the starting surface backwards, starting inside the iPhone body, and then extrudes it, by a thickness of 8mm outwards. The reason for this is, because currently Blender 2.5 does not have the weight edge feature included and thus the Bevel modifier would also chamfer the edge of the rim which lies on the iPhone body.

3D WORKSHOP : Modeling an iPhone 4 with Modifiers





To finish the basic iPhone body, it can be duplicated and moved to another layer. The mirror and solidify modifiers are applied and missing faces are built into the mesh to produce a water tight body. As the screenshot shows, the faces for the future display panel for texturing is built in.

Step 3: Interactive Detailing

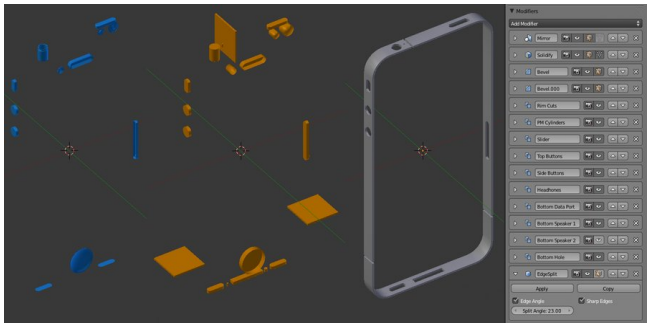
At this stage, detail meshes can be added interactively with the use of the Boolean modifier.

This approach has a certain advantage compared to hand modeling geometry such as an opening for a button insert. Through using the Boolean modifier, the designer can place the geometry which should be subtracted and the modifier will perform this tasks automatically. Rather then redoing or adjusting the mesh by hand, there is more time to focus on fine tuning the position, scale, rotation, and geometry of the Boolean mesh.

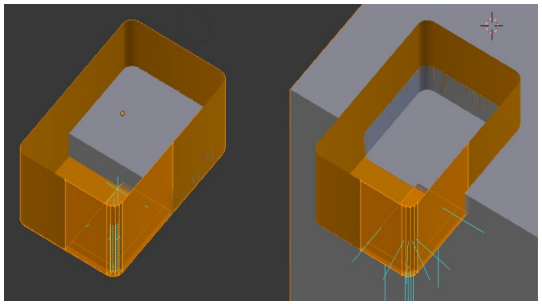
For the Boolean modifier and others to work as desired certain restrictions have to be paid attention to. In general the first rule is the order of tasks being performed. The following image shows on the far right side the advisable order of modifiers for the rim structure. At first is the Mirror modifier which produces a full thin surface, followed by the Solidify modifier which creates the thickness. If the Mirror modifier would be second, it would also mirror inner faces created by the Solidify modifier and those bad geometry elements are often the reason why the Boolean modifier would produce bad results.

It is important to add the Bevel modifier at this stage before we start utilizing the Boolean modifier, because only the outer edge of the rim should be beveled. If the Bevel modifier was last it would also be applied to all edges resulting from the Boolean task. Also all Boolean meshes have Smooth Shading turned on as well as the Rim. With the EdgeSplit modifier at the end of the modifier list, each object has smooth rounded surfaces and creased sharp edges automatically in 3D view and at Render Time

The orange elements are those being utilized for the Boolean operation. As a second rule, it is very important here to have clean and



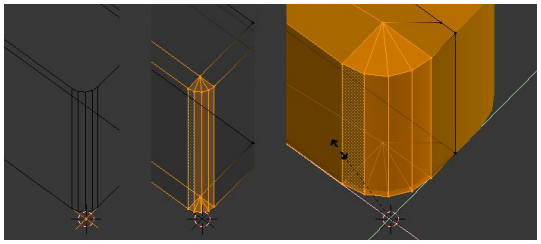
simple mesh models without any duplicated faces or other issues, as they will produce problems for the Boolean modifier as well. Water tight volumes are not needed and as can be seen some objects are simple surfaces. However their surface normal defines the direction the Boolean operation works - thus is it important to make sure that they all uniformly face outside when the object is being used as a Boolean Difference (which subtracts geometry). The following composition explains this concept.



The concept of using the modifiers for mesh modeling lies in the interactive ability to change position and also shape and thus rapidly fine tune the design. The following

3D WORKSHOP : Modeling an iPhone 4 with Modifiers

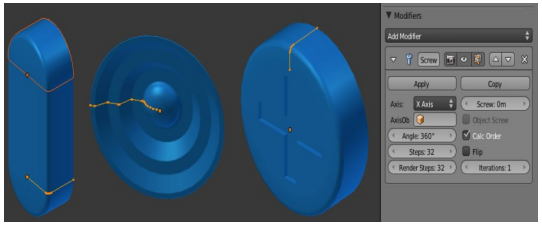
screenshot displays a quarter model of Boolean object. Starting with the left, a small mesh cross aids as a corner position for the 3D cursor which can be snapped to the cross through the [Shift +S] Snap menu and selecting the cursor to Selection option. The right image shows the edge geometry being scaled with using the 3D cursor as the pivot point and making use of the Z axis exclusion. [s + shift + z] This method currently works very well when corner rounding might change during the design process.



This set of Boolean modifiers for the rim model can already slow down the screen refresh. Each time one object is modified Blender will recalculate the complete modifier chain which explains the increasingly longer computation time. To speed up editing it is advisable to only have those modifiers visible which are being worked on by clicking onto the “eye” icon. During render time the outcome will still be visible.

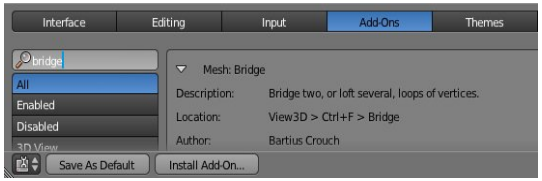
The blue elements on the left are representing the detail mesh objects, such as buttons, sliders, camera, and speakers. They are built based on the very same basic Boolean objects keeping the mesh density identical to guarantee a tight fit and prevent open cracks. In the case where a button has to be smaller than the Boolean, the profile has to be scaled down. The Solidify modifier cannot produce just an offset surface as Blender does not offer a Scale modifier. Also because Blender 2.5 does not yet support edge weighting which prevents selective edge beveling through the modifier, a different approach was used to evenly build all buttons with nicely rounded edges. The new Screw modifier is a great help to quickly produce lathe / revolve like objects by only using a base profile.

The following screenshot displays the different application areas.



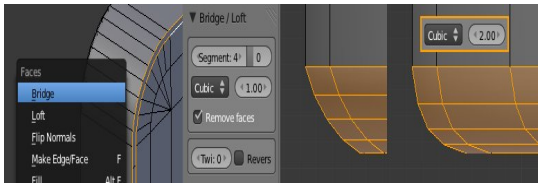
Through duplication and slight modification different buttons with altered size and proportions can be generated quickly. The Screw modifier allows you to specify the amount of rotation which is 360 degrees for the two right objects and 180 degrees for the left one. Selecting the calculate check box also makes sure that no normals are flipped in the resulting mesh.

Another very useful tool in Blender 2.5 is Bartius Crouch’s Loft and Bridge script. Not all Blender builds include it yet and it can be installed through the Install Add-On function in the User Preferences. The script can be downloaded from the posted link in the resources section.



The previous image shows two edges being selected. To quickly access the bridge script [ctrl + f] can be used, which opens the Faces dialog. The great thing about this script is that it does not only do a basic linear fill between two edges, but can interpolate between them, which means that the curvature can be controlled and smooth bevels can easily be constructed.

The Cubic interpolation as shown in the middle image has to be selected and the strength value defines how far the created bridge mesh will blend into the start and end geometry. As shown, the far right image presents a much smoother flow from the gray faces into the orange rounded mesh. This is mainly a function found in NURBS application and it is a great pleasure to see this in Blender now.



3D WORKSHOP : Modeling an iPhone 4 with Modifiers

Final Model: Preparation for Rendering

For rendering and scene design it is advisable to bake all geometry generating modifiers such as Boolean, Screw, Mirror, Solidify into touchable mesh. The are many reasons for this. The Boolean modifier can slow down the screen refresh. This modifier in 2.5 also ignores the material of the subtraction object. The Boolean result can also show some surface normal issues as visible in the left quick rendering around the home button. In addition, all objects have to be moved when repositioning the iPhone including all Boolean meshes. This can quickly generate a lot of unneeded overhead.

Screw, Solidify, and Mirror modifiers also do not really allow UV texturing of those meshes since it is impossible to unwrap them. Those meshes are generated on the fly.

Keeping a back-up geometry set on another layer might in general be a good idea, or having one Blend file for the modeling, and one Blend file for rendering. In this case it also makes sense to properly name all the objects, thus making it easier when a specific element has to be imported from a modeling file into the rendering scene. For further details, the supplied Blender file can be consulted, which contains the different steps separated through layers.

Closing Comments:

Very often the "Blender 2.5 vs 2.4" question is asked at the Blenderartist forum. Blender 2.5 possesses, compared to Blender 2.4, many enhancements which significantly increase productivity. While a few tools from 2.4 are still missing the majority of new additions fully justify a switch to the new architecture. This affects the cleaner interface, the new continuously growing add-on system, the more logical workflow and efficiency possible with Blender 2.5.

In particular can this be observed through Blender being currently considered even more as a host application for open source projects in the industry ranging from simple modeling tasks to quite complex visualisation projects such as the BlendME add-on, which connects Blender with the opens source products like openFOAM and others ●

References:

Apple Website for iPod, iPhone, iPad

http://developer.apple.com/programs/mfi/cases.html

Apple iPhone 4 Dimensional Drawing

http://devimages.apple.com/programs/mfi/dimensions/iPhone-4-dimensions.pdf

Apple iPhone Gallery

http://www.apple.com/iphone/gallery/#gallery01

Bartius Crouch Loft Bridge Script Add-on:

http://sites.google.com/site/bartiuscrouch/scripts/bridge

Demo File:

Blender file with all steps spread over the layers.

by - **Stephen Bates**

Introduction

Have you ever wondered if Array modifier can be used to create plants, well here is how to do it.

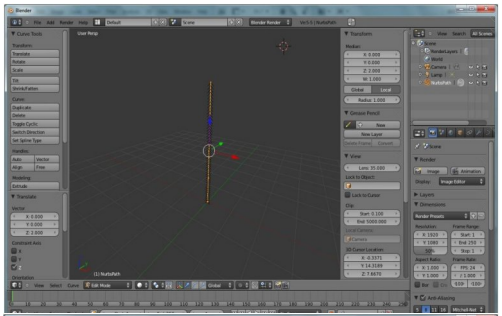
Step1. In blender delete the cube and add a Path curve, rename it to Stem_Path.

Step2. Tab into edit mode rotate it -90 Degrees in Y and move it up 2 units in Z.

Step3. Tab back out of edit mode.

Step4. Now add a cylinder, set the vertices to 6, radius to .5, depth to 1 and un-tick Cap Ends.

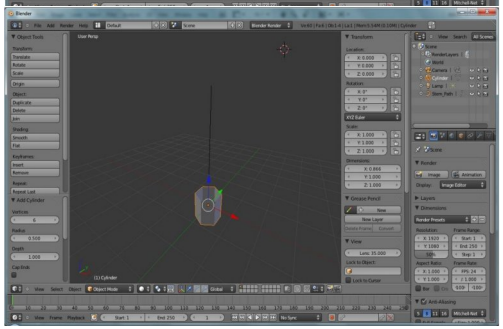
Step5. When done rename it to Stem.



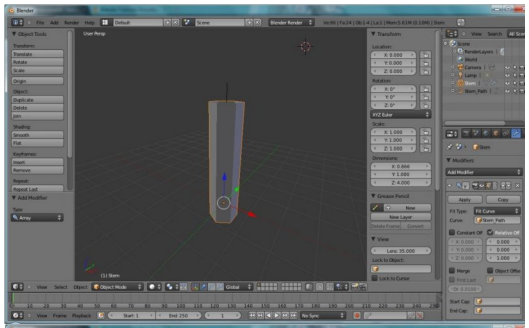
Step6. Add a modifier to the Stem, change Fit Type to Fit Curve and select the Stem_Path in the Curve Field.

Step7. In the Relative Offset, set X to 0 and Z to 1.

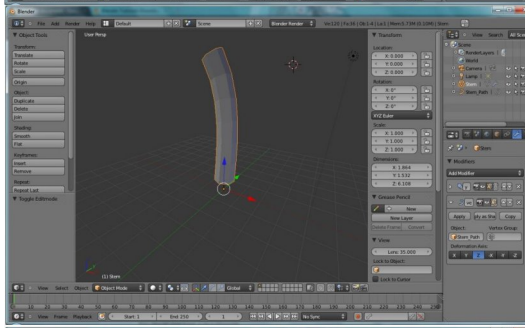
Step8. The stem should now be the length of the Stem_Path.



Step9. Now Add a curve modifier to the Stem, set the object to Stem_Path, and the Deform Axis to Z.

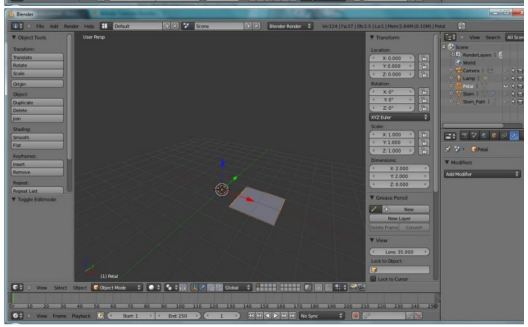


Step10. You should now be able to select the Stem_Path and in edit mode grow the stem to the desired length.



Step11. Now select everything in the scene with the A key, and press H to hide.

Step12. Now create a plane, Rename it to Petal, tab into edit mode and move it 2 units in X.



Tab back into object mode.

Step13. Create an Empty, rename to PetalArc.

Step14. Select the Petal, and add an array.

Step15. Set the count to 4, un-tick Relative Offset, tick Object

3D WORKSHOP : Creating Plants Using Arrays

Offset and select the PetalArc in the field below.

Step16. Select the PetalArc Empty and rotate in Y -22 deg, rotate in Z 15 Degrees and scale XYZ to 0.9.

Step17. Create another empty and rename it to PetalRotate and rotate it 45 Deg in Z.

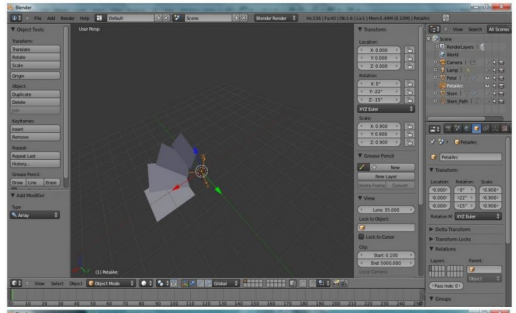
Step18. Add an ico sphere, rename it to Stamen and move it to one side.

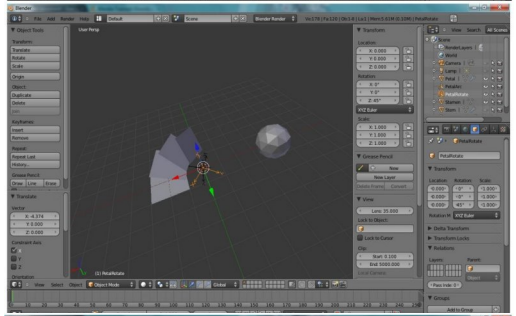
Step19. Select the Petal again, and add another array, set the count to 8.

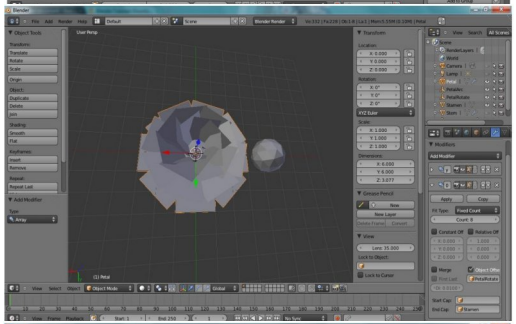
Step20. Untick Relative Offset, tick Object Offset and select the PetalRotate below. In the EndCap Field select the Stamen.

Step21. You should now have a nice blossom.

Step22. Hide everything again, and in the scene window (top right) click the eye next to the stem and Stem_Path to make them Visible.



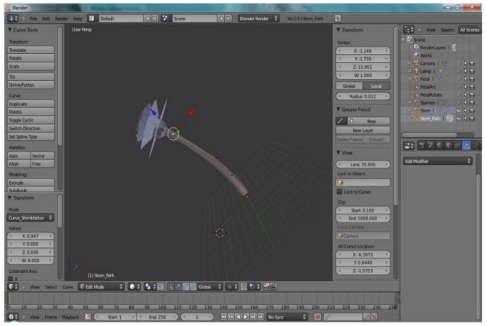




Step23. Select the stem, then in the stems modifier list, select the Array and set the End Cap to Petal.

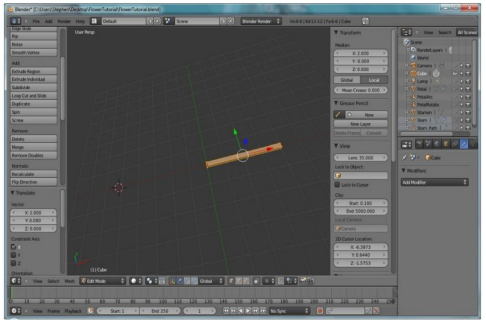
Step24. You should now have a Stem with a Flower, select the Stem_Path, tab into edit mode and select the second vertex down from the top, (you dont want to select the top vertex).

Step25. Now press ALT+S and scale the vertex down, or change its radius (same thing), I've set mine to 0.012.



Step26. Then select the top two vertices, and make the plant a little bigger by dragging up in Z.

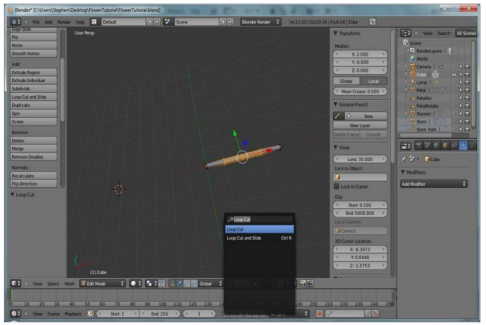
Step27. Hide everything again then create a cube.



Step28. Tab into edit mode, hit S then type .1, this will make the cube one-tenth of its original size.

Step29. Now hit S X 20, to scale it 20 times bigger in X, then G X 2 to move it 2 units across.

Step30. Hit space and type Loop cut, and using the mouse wheel, create 3 cuts so you have 4 equal segments.



3D WORKSHOP : Creating Plants Using Arrays

Step31. Now select 4 edges, hit E to extrude, and right click without moving the edges.

Step32. Now hit S Y to scale in Y and create the leaves.

Step33. You can just apply a texture with an Alpha, or make them more leaf shaped.

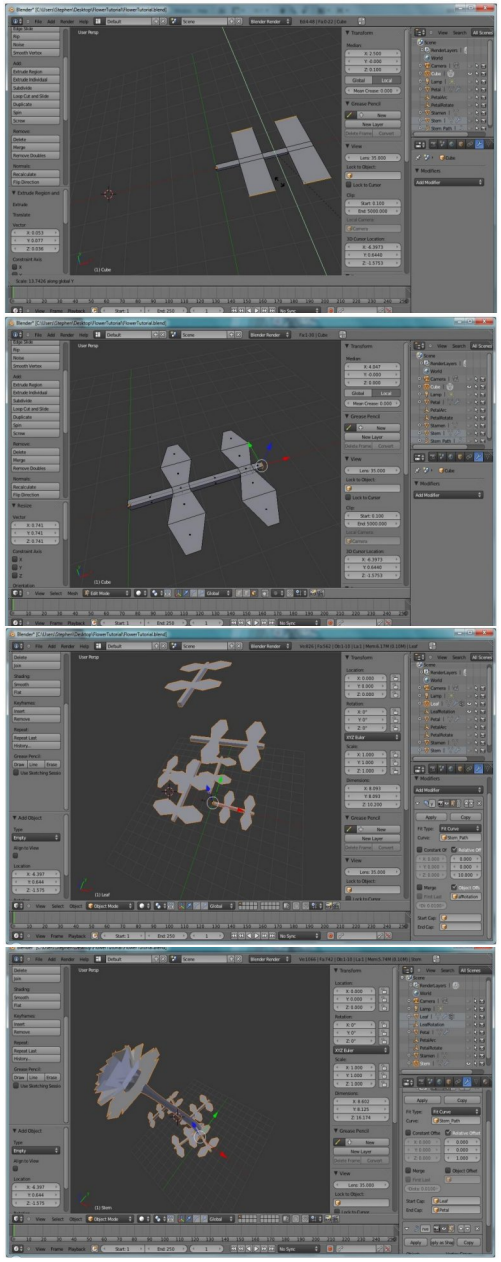
Step34. Tab back into Object mode, and rename the cube to Leaf.

Step35. Now create another Empty, rename it to LeafRotation and rotate it in Z 90 degrees.

Step36. Select the leaf again, and in the modifier list add an array, set the relative offset for X and Y to 0 and set Z to 10.

Step37. Tick Object Offset, and select LeafRotation in the field below, and you should have the following.

Step38. Now Hide everything once more, and un-hide the Stem.



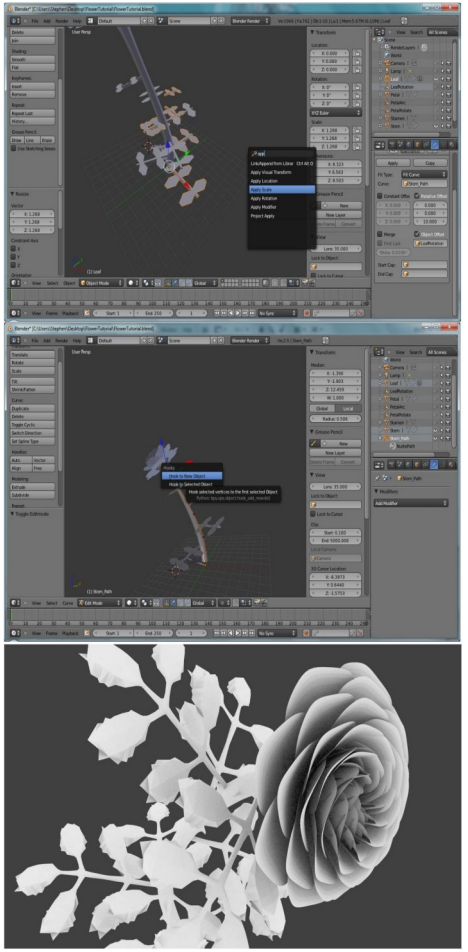
Step39. Select its existing array, and in the Start Cap field, select Leaf, the plant will now have leaves following the stem.

Step40. If you want the leaves to be bigger, select the leaf object, scale it then you need to apply the scale.

Step41. You won't see the changes until you select the stem, tab into edit mode, then tab back out.

Step42. Once you are happy with the flower, make the stem path Visible, select the top 2 vertices.

Step43. Press CTRL+H and hook to new object, you can now go back into object mode, and animate the plant with the new empty that the hook created.



Here is one I took a bit more time over, for the leaves, instead of adding them to the startcap, I created a vertex group for the thin bit that connects the leaves, and applied a cloth modifier with the leaf pinned to that vertex group, I then used the curve modifier, and selected the StemPath ●

BLENDER STUFF : CanTree - Online Tree Generator

by - **Arnaud Couturier**

Introduction

I am Arnaud Couturier, from New Caledonia, in the Pacific Ocean. I use the pseudonym Piiichan in the online community. My passion lies in art, programming, and the combination of both, such as computer games and digital content generators.

One of my latest experiments is CanTree, a free online tree generator. Before I go any further, you can test it right now if you wish, simply go to http://arnaud.ile.nc/cantree.

With Cantree, you can generate 2d trees, that look almost like 3d, because of simulated ambient occlusion. You have some degree of control on the aspect of the tree. For example you can easily change the size and complexity of the tree using sliders. You can also choose your leaf texture(s), wood texture, and flower textures. You can even provide your own textures, and this way you can include anything in your tree, such as fruits, photos of yourself, or any image you want.

So far, there's nothing particularly new about this tree generator, most people have used at least one (possibly excellent) tree generator at some point. The uniqueness of CanTree is that the trees are generated "online" by your web browser. There is no plugin, installation or download required. It's also free, and under the MIT licence, so CanTree must be one of the most accessible tree generators one can find.

CanTree stands for canvas tree, I am referring to the HTML 5 canvas element. HTML 5 is the set of standards for the new web. Canvas is one of them, and one of the most exciting, because it makes possible the drawing of impressive 2d and 3d graphics right inside the browser, without any plugin such as Adobe Flash. Most major web browsers, namely the latest versions of Firefox, Chrome, Safari, and Opera now support canvas. Only Internet Explorer (IE) doesn't support it, so CanTree simply won't work in IE. Opera is also showing some bugs. I haven't tried the generator with the latest beta version of IE 9, which as far as I know should support canvas.

You can save your trees as transparent png image files. Some people have been fooled by the "3D feeling" of the trees, and have requested .obj export, but the trees are in 2d, not 3d. The only other possible export format would be a vector format, such as SVG. It would allow a complete control over each part of the tree, useful for interactive animations (falling leaves, branches moving in the wind...) to be drawn in real time in a web page. However, a tree is made of tens of thousands leaves and branches, so redrawing them constantly at a decent frame rate (at least 25 frames per second) would require substantial computing power.

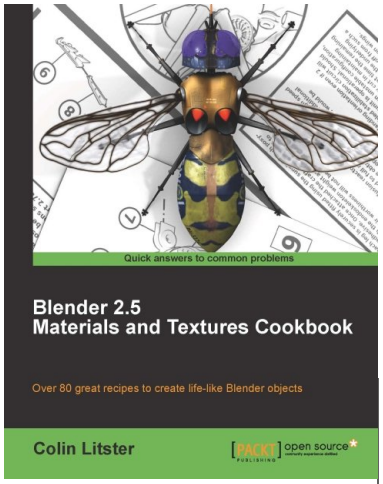
In the future, I plan to improve the control the user has over the shape of the tree, to allow much more varied tree shapes such as palm trees for example. This way, you could choose the species of your tree, choose the size, complexity and image resolution, then get as many different trees as you want, but only for the species you're interested in. I also have planned forces that would alter how the tree grows, like wind and gravity. Other possible features include the addition of creepers and roots. I've started working on these features, but I can't tell when I'll be done, since CanTree is developed during my free time, as a hobby.

So, grab your web browser and have a look at http://arnaud.ile.nc/cantree. Keep in mind that Internet Explorer can not run the generator, and that Opera is still buggy ●

To learn more about HTML 5, I recommend "Dive Into HTML 5" by Mark Pilgrim at http://diveintohtml5.org. For a summary of which parts of HTML 5 are supported in all major web browsers, check http://www.findmebyip.com/litmus.

For bug reports and comments, leave a message on the blender artist thread http://blenderartists.org/forum/showthread.php?t=205000&page=1

BLENDED REVIEW : Blender 2.5 Materials and Textures Cookbook



by - **Sandra Gilbert**

★★★★★

Blender 2.5 Materials and Textures Cookbook

by: Colin Litster

312 pages

Publisher: Packt Publishing

Psst, hey you... come here ... I have a secret... closer.... closer.... Too Close! Back up.

Okay now that I have your attention...

Materials can make or break your image or animation.

I know, we should all know that by now, but just because we know it doesn't mean we can successfully apply it. Great materials can be difficult. First you have to understand all the unique properties of the material you are trying to create. Then you have to figure out how to make it or even fake it in Blender. Blender has a powerful set of material and texturing tools and options. With such power of course comes more than a fair share of confusion on just how to use those tools effectively.

Colin Litster has taken on the challenge of explaining Blender's material and texturing system in his new book, Blender 2.5 Materials and Textures Cookbook.

Using a cookbook style format, Colin provides a "recipe" of clear steps needed to create each material. After the "recipe", Colin then provides an easy to understand explanation of the material. He covers why each option and setting was used and how it contributed to the overall material. The Blender 2.5 Materials and Textures Cookbook contains nine chapters that cover a nice selection of both natural and man made materials as well as some nice special effects and animated materials.

In addition to creating materials, Colin devotes an entire chapter to managing all those materials. He covers setting a default scene for material creation through naming conventions and appending/linking. All of which are important aspects to maintaining a dynamic personal database of blender materials.

Conclusion

Colin has created an invaluable resource for anyone seeking to improve their knowledge and understanding of Blender's Materials and Texture system ●

From the Packt Publishing Website about Blender 2.5 Materials and Textures Cookbook

You will learn how to emulate the reflective properties of natural materials and how to simulate materials such as rusted iron, which is difficult to make believable. Transparency and reflection are both tricky natural surface properties to simulate but these recipes will make it easy. Explore ways to speed up animations by using special painting techniques to significantly lower render times. By the end of the book, you will be able to simulate some of the most difficult effects to recreate in any 3D suite, such as smoke, fire, and explosions.

What you will learn from this book :

Understand the new Blender 2.5 user interface that simplifies creation of materials and textures

Explore the complex task of UV mapping of a human face

Use the Sub Surface Scattering commands in Blender to create objects the way you want

Confidently simulate materials such as smoke, flames, and explosions using the Blender 2.5 Smoke Physics module

Create an entire ocean that animates and reacts with objects in the water by using the new Blender 2.5 features

Employ simple repeating textures that can be applied with infinite variety without appearing to repeat

Synthesize complex materials without complex mesh objects by using alpha transparency

Create incredible moving textures and materials by using Blender 2.5 animation curves

Create flexible materials that can curve around corners or apply themselves to complex winding meshes without unwanted texture distortion

Manage Blender 2.5 materials and textures and effectively apply them to your Blender projects

Want to write for BlenderArt Magazine ?

Here is how

Step1. Choose what you want to write.

- Tutorials explaining Blender features, 3dconcepts, techniques or articles based on the focused theme of the issue.

- Reports on useful Blender events throughout the world.

- Cartoons related to blender world.

Step2. Send submissions to sandra@blenderart.org.

- Send us a notification on what you want to write and we can follow up from there.

Step3. Some guidelines you must follow.

- Images should be properly cut and represent the text appropriately.

- Images should be provided seperately in a folder named (images, img or pictures).

- Images should be named/labled likewise (image1 or img1 etc).

- Provide proper captions for images if and when needed.

- Image format prefered is PNG but good quality JPG can also do.

- You can submit inline images in documents like DOC or Openoffice ODT etc but make sure the images were properly names before importing them in docs.

- Images inside a PDF are a strict no, but a pdf document with images if provided to show how the author wants the formating of doc will be appreciated.

- Make sure that screenshots are clear and readable and the renders should be at least 800px, but not

- Text should be in either ODT, DOC, TXT or HTML.

Step4. Archive them using 7zip or RAR or less preferably zip.

Step5. Additional stuff that you can do

- Please include the following in your email:

- Name: This can be your full name or blenderartist avtar.

- Photograph: As PNG and maximum width of 256Px. (Only if submitting the article for the first time)

- About yourself: Max 25 words .

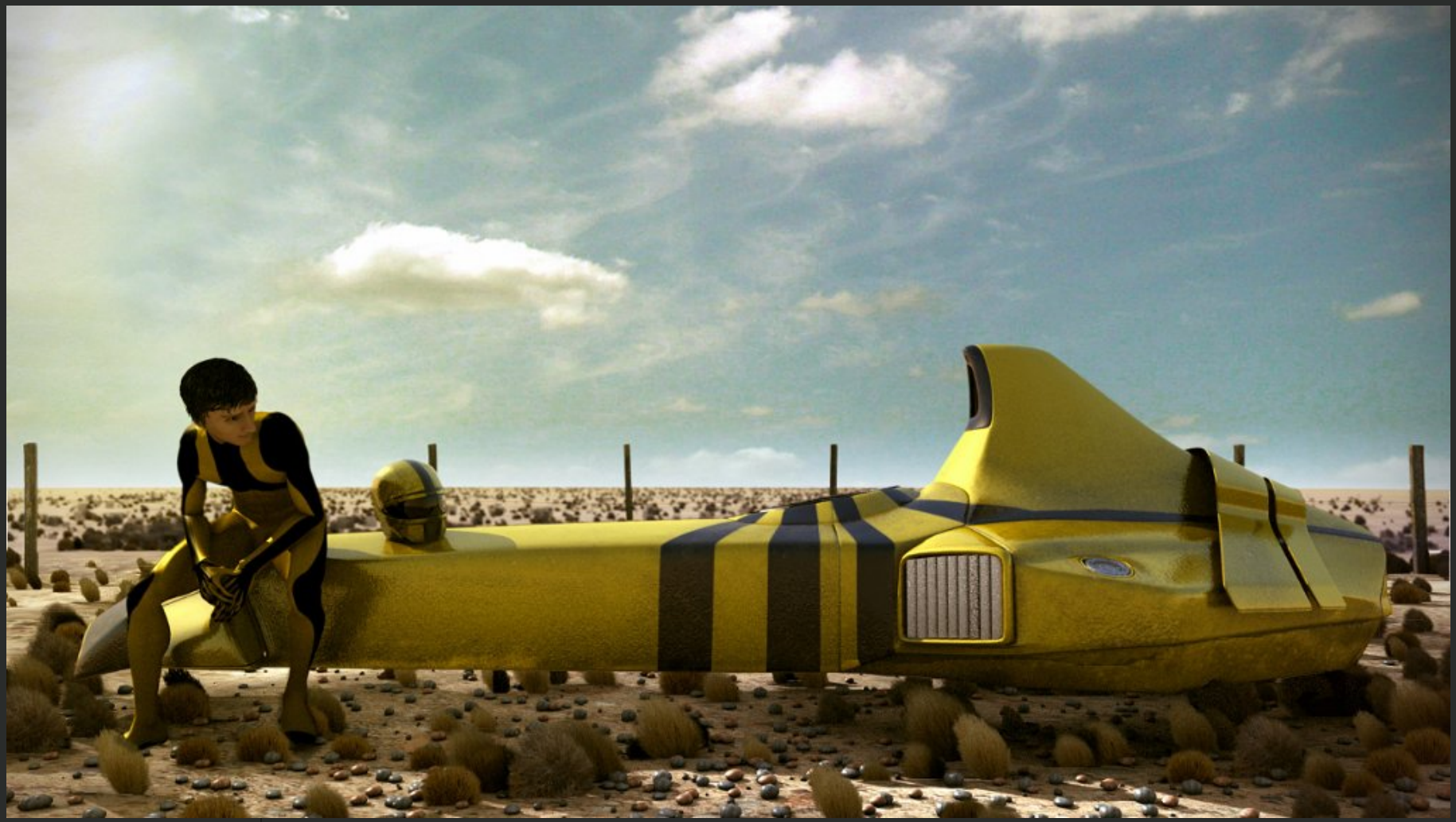
- Website: (optional)

Note: All the approved submissions can be placed in the final issue or subsequent issue if deemed fit. All submissions will be cropped/modified if necessary. For more details see the blenderart website.

BA takes no responsibility fo the material in any form and the submission will automatically mean that you have agreed to the blenderart terms and conditions for submission for more information please do read the disclaimer.

Poissons sous la pluie - by Sebastien Mornat

F1 - by Luis Roach



Zoe - by Richard Rabassa

Forest - by Thomas Kristof